International Journal of Aviation Psychology, 9(2), 125-138.

MODES IN HUMAN-MACHINE SYSTEMS: REVIEW, CLASSIFICATION, AND APPLICATION

Asaf Degani
San Jose State University, CA
and NASA Ames Research Center
Moffett Field, CA

Michael Shafto
NASA Ames Research Center
Moffett Field, CA

Alex Kirlik
Department of Industrial and Systems Engineering
Georgia Institute of Technology
Atlanta, GA

ABSTRACT

This article surveys and discusses human interaction with automated control systems that employ modes. We focus our discussion on the features of the control system that lead to mode confusion and error. We first provide working definitions of the term "mode" and discuss key constructs that contribute to mode error, such as mode ambiguity and user's expectations. Following this, we discuss human interaction with automated control systems, in general, and cockpit automation, in particular. To provide a formal description of human interaction with such control systems, we introduce a modeling language, Statecharts, which is used by system engineers to specify complex control systems. We use the Statecharts language to describe the three types of modes that are commonly found in modern control systems: interface, functional, and supervisory modes. Examples from cockpit automation are used to illustrate each type of mode. The paper concludes with a brief discussion of the links between the mode constructs and formal representation of human interaction with control systems.

INTRODUCTION

> Mode: (1) manner of acting or doing; method; way. (2) the natural disposition or the manner of existence or action of anything: form." (Source: Latin. modus—measure or manner.) *Webster's Dictionary*, 1994.

Much attention has recently been devoted to the issue of human interaction with complex, dynamic, and highly automated control systems. These complex control systems are found on board aircrafts, ships, and in military command-and-control centers. The human factors research community has learned from field studies and empirical investigations that certain aspects of automatic control systems are problematic to users (Billings, 1996). Three factors are commonly cited: (1) The interface to the operator (e.g., pilot) may not always provide adequate information about the configuration of the machine; (2) intricate and confusing interactions may exist among the many components of a complex machine; (3) the operator has an insufficient (mental) model of the machine's behavior (Woods, Sarter, & Billings, 1997).

These factors may limit the operator's ability to anticipate the future behavior of the machine. That is, following either a manually triggered event (e.g., setting a parameter), or an automatically triggered event (e.g., reception of an electronic signal), the operator may be *unable* to predict the next configuration of the machine. The user's inability to resolve the next configuration of the machine leads directly to confusion and error.

In modern control systems, a mode is a common architecture for grouping several machine configurations under one label. The set of modes in a control system corresponds to a set of unique machine behaviors. The operator interacts with the machine by switching among modes manually, or monitoring the automatic switching triggered by the machine.

However, our understanding of modes and their potential contribution to confusion and error is still far from complete. For example, Johnson & Engelbeck (1989) demonstrated that there is widespread disagreement among user interface designers and researchers about what modes are, independent of how they affect users. This blurred vision, found not only in the human-computer-interaction domain, impedes our ability to develop methods for representing and evaluating human interaction with control systems. This limitation is magnified in high-risk systems such as automated cockpits (Abbott, Slotte, & Stimson, 1996), for which there is an urgent need to develop methods that will allow designers to identify, early in the design phase, the potential for error.

In the ensuing discussion, our focus is on the unique features of the machine that lead to confusion and error. Other important factors such as human attention and perception, as well as display layout and coding of information, are beyond the scope of this paper. The discussion is set in the context of human interaction with computers and devices, in general, and pilots' interaction with modes of the Automatic Flight Control System, in particular.

MODES: DEFINITIONS AND CONSTRUCTS

Mode Behavior

One of the first treatments of modes came from the science of cybernetics—the comparative study of human control systems and complex machines. Ashby (1956/1964) set forth the construct of a machine with different behaviors. The following is a simplified description of Ashby's construct: A given machine may have several components (e.g., X1, X2, X3). For each component there is a finite set of states. On *Startup*, for example, the machine initializes itself so that the active state of component X1 is *a*, X2 is *f*, and X3 is *k* (see Table 1). The vector of states (a, f, k) thus defines the machine's configuration on *Startup*. Once a built-in test is performed, the machine can move from to *Startup* to *Ready*. The transition to *Ready* can be defined so that for component X1, state *a* undergoes a transition and becomes *b*; for X2 state *f* transitions to *g*, and for X3, *k* changes to *l* . The new configuration (b, g, l) defines the *Ready* mode of the machine. Now, there might be a third set of transitions, for example, to *Engaged* (c, h, m), and so on.

Table 1. A Machine With Different Behaviors

	X1	X2	X3
Startup	a	f	k
Ready	b	g	l
Engaged	c	h	m

The set of configurations labeled *Startup*, *Ready*, and *Engaged*, if embodied in the same physical unit, corresponds to a machine with three different ways of behaving. A real machine whose behavior can be so represented is defined by Ashby as a machine with *input*. The input triggers the machine to change its behavior (Ashby, 1956/1964, Chap. 4).

One source of input to the machine is manual—i.e., the user selects the mode (e.g., by turning a switch), and the corresponding transitions take place. But there can be another type of input: If some other machine selects the mode, the input is "automatic." More precisely, the output of the other machine becomes the input to our machine. For example, a separate machine performs a built-in test and outputs a signal that causes the machine in Table 1 to transition from *Standby* to *Ready*, automatically. In Ashby's terminology, the two machines are *coupled* (p. 48).

For the purposes of this discussion, we define a mode as a machine configuration that corresponds to a unique behavior. This is a very broad definition of the term. But later in this paper we will constrain this definition from our special perspective: user interaction with control systems that employ modes.

Mode Error and Ambiguity

Donald Norman, in his 1981 paper "Categorization of Action Slips," coined the term "mode error" (but see also Reason & Mycielska, 1982). In Norman's scheme, mode errors fall into a class of errors that involve forming and carrying out an intention. That is, when a situation is falsely classified, the resulting action may be one that was intended and appropriate for the perceived or expected situation, but inappropriate for the actual situation (p. 6).

Nevertheless, there remains an open question as to what kind of situations lead to mode error. Thimbleby (1982) addressed this issue by studying the design of modes in a word processor. Specifically, he looked at situations in which the user's input has different interpretations, depending on the mode. For example, in one word processing application, the keystroke *d* can be interpreted as either (1) the literal text "d," or (2) as the command "delete." The interpretation depends on the word processor's active mode: either Text or Command.

Similar examples exist in cockpit automation. For example, there are two types of speeds in the Automated Flight Control System (AFCS): Mach and Indicated Air Speed. Rotating the speed knob on the interface can be interpreted as a new Mach value or new Indicated Air Speed value, depending on the active mode. This type of design, in which the same user input elicits two different interpretations, is defined by Thimbleby as mode ambiguity.

User Expectations

Monk (1986) linked mode ambiguity to mode error by interjecting the notion of user expectations. In his view, "mode ambiguity" will result in mode error only when the user has a false expectation about the result of his or her actions (p. 314). Monk makes a distinction between two types of ambiguity—one that leads to mode error, and one that doesn't. An example of mode ambiguity that does lead to mode error is a time-sharing operating system in which keystrokes are buffered until the "Return" or "Enter" key is pressed. However, when the buffer gets full, all subsequent keystrokes are ignored.

This feature leads to two possible outcomes—all or only a portion of the keystrokes will be processed. The two outcomes depend on the state of the buffer (not full, full). Since the state of the buffer is unknown to the user, false expectations may occur. The user's action—hitting the "Return" key and seeing only part of what was keyed on the screen (because the buffer has already filled up)—is therefore a "mode error."

An example of mode ambiguity that does *not* lead to mode error is a common end-of-line algorithm which determines the number of words in a line. An ambiguity is introduced because the criteria for including the last word on the current line, or wrapping to the next line, are unknown to the user (e.g., a four-letter word may stay on the current line, but a five-letter word may be wrapped automatically to the next line).

Nevertheless, as long as the algorithm works reasonably well, the user will not complain. Why? Because he or she has not formed any expectation about which word will stay on the line or scroll down, and either outcome is usually acceptable. Therefore, mode error will not take place, even though mode ambiguity does indeed exist.

It remains, however, an open question as to how one can reliably predict user expectations (Monk, p. 314). In the following sub-sections we extend the above constructs by discussing three elements that can be used to evaluate user expectations: (1) the user's task, (2) the user's knowledge, and (3) the user's ability to sense inputs that prompt mode transition (Degani, 1996).

User Tasks

One important element that constrains user expectations is the task at hand. If discerning between two or more different machine configurations is not part of the user's task, mode error will not occur. Consider, for example, the radiator fan of your car. Do you know, *always*, what configuration (*Off*, *On*) it is in? The answer, of course, is no. There is no such indication in most modern cars. The fan mechanism changes its mode automatically depending on the output of the water temperature sensor. Mode ambiguity exists because at any point in time, the fan mechanism can change its mode or stay in the current mode.

The configuration of the fan is completely ambiguous to the driver. But does such mode ambiguity lead to mode error? Not at all, because monitoring the fan configuration is not part of the driver's task. Therefore, the user's task is an important determinant of which machine configurations must be tracked by the user and which machine configurations need not be tracked (and therefore need not be indicated on the display).

User Knowledge

A second element that is part of the assessment of user expectations is user knowledge about the machine's behaviors. By this, we mean that the user constructs some mental model of the machine's "response map" (but see also other treatments of this issue in the mental models literature—Gentner & Stevens, 1983; Stenning & Oberlander, 1995). This mental model allows the user to track the machine's configuration, and most importantly, to anticipate the next configuration of the machine. Specifically, our user must be able to predict what the new configuration will be following a manually triggered event or an automatically triggered event.

The problem of reliably anticipating the next configuration of the machine becomes difficult when the number of transitions between configurations is large. Another factor in user knowledge is the number of conditions, that must be evaluated as TRUE, before a transition from one mode to another takes place (Mett, Crowe, & Strain-Clark, 1994, pp. 96, 243, 295). For example, the Automated Flight Control Systems of modern aircraft can execute a fully automatic (hands-off) landing. Several conditions (two engaged autopilots, two navigation receivers tuned to the ILS frequency, identical course set, and more) must be TRUE before the aircraft will execute automatic landing.

Therefore, in order to reliably anticipate the next mode configuration of the machine, the user must have a complete and accurate model of the machine's behavior, including its configurations, transitions, and associated conditions. This model, however, does not have to be complete in the sense that it describes every configuration and transition of the machine. Instead, as discussed earlier, the details of the user's model must be merely sufficient for the user's task—a much weaker requirement.

Sensing of Triggering Events.

The third element in the assessment of user expectations is the user ability to sense the condition(s) that trigger a transition. Specifically, the user must be able to first sense the events (e.g., Flight Director is engaged; aircraft is more than 400 feet above the ground) and then evaluate whether or not the transition to a mode (e.g., Vertical Navigation) will take place. These events are usually made known to the user through an interface. Nevertheless, there are more than a few control systems in which the interface does not depict the necessary input events. Such interfaces are said to be incorrect (Degani & Heymann, forthcoming).

In large and complex control systems, the user may have to integrate information from several displays in order to evaluate whether the transition will take place or not. For example, one of the conditions for a fully automated landing (autoland) in a two-engine jetliner is that two separate electrical sources must be online, each one supplying its respective autopilot (two autopilots are necessary for executing an autoland). This information is external to the Automatic Flight Control System, in the sense that it involves another aircraft system (electrical). The user's job of integrating events, some of which are located in different displays, is not trivial. One important requirement for an efficient design is for the *interface* to integrate these events and provide the user with a succinct cue.

In summary, we have discussed three elements that help to determine whether a given mode ambiguity will or will not lead to false expectations. First is the relationship between mode ambiguity and the user's task. If distinguishing between modes (e.g., radiator fan *On* or *Off*) is not part of the user's task, no meaningful errors will occur. Second, in cases where mode ambiguity *is* relevant to the user's task, we assess the user's knowledge. If the user has an *in*accurate and/or incomplete model of the machine's response map, he or she will not be able to anticipate the next configuration and mode confusion will occur. Third, we evaluate the user's ability to sense input events that trigger transitions. The interface must provide the user with all the necessary input events. If it does not, no accurate and complete model will help—the user may know what to look for but will never find it. As a result, confusion and mode error will occur.

The constructs described above are measurable aspects of human interaction with machines. As such, they can be used to form the foundation of a systematic and quantitative analysis, as we shall briefly discuss in the Conclusion section. But before such an analysis can be undertaken, some form of representation is needed. In the next section we shall discuss both a classification scheme for the type of modes found in complex control systems, and a modeling language, Statecharts, that can be used to represent human interaction with these systems.

REPRESENTATION AND CLASSIFICATION OF MODES

We continue by considering and elaborating on the basic definition of *mode* from our own perspective: user interaction with automated control systems. To this end, we propose a classification that encompasses three types of modes: (1) *Interface* modes that specify the behavior of the interface, (2) *Functional* modes that specify the behavior of the various functions of a machine, and (3) *Supervisory* modes that specify the level of

user and machine involvement in supervising the process. Before we proceed to discuss this classification, we shall briefly describe a modeling language, Statecharts, that will allow us to represent these modes.

Statecharts

It is well recognized, in both the research literature and in practice, that a State Machine model is a natural medium for describing the behavior of a mode-based system (Jacob, 1983; Sherry & Ward, 1995). A basic fragment of such a description is a *state transition* which captures the states, conditions, and transitions in a system, e.g., "while the aircraft is in Cruise mode (current state), and button x is pressed and the descent profile is armed (two conditions), the aircraft enters Descent mode (new state)."

The Statecharts language is a visual formalism for describing states and transitions in a modular fashion by extending the traditional Finite State Machine to include three unique features: *hierarchy*, *concurrency,* and *broadcast* (Harel, 1988). *Hierarchy* is represented by sub-states encapsulated within a super-state. *Concurrency* is shown by means of two (or more) independent processes working in parallel. The *Broadcast* mechanism allows for coupling of components, in the sense that an event in one end of the network can trigger transitions in another. These features of Statecharts will be further explained in the following examples.

Interface Modes

In the Human Computer Interaction (HCI) literature, most authors consider modes a method for changing the format of the interface. The user's input to the computer is interpreted differently according to the active mode (Poller and Garter, 1984; Tesler, 1981; Thimbleby, 1990, Chap. 11). Consider, for example, the Mode Control Panel (MCP) of a Boeing B-757. The MCP is the interface through which the pilots interact (e.g., by commanding mode changes and setting parameters) with the Automatic Flight Control System of the aircraft. Specifically, we will discuss here the "Speed Intervene" function and associated knob. While the fully automatic Vertical Navigation (VNAV) mode is active, the speed parameter is obtained from the Flight Management Computer, which computes the most economical speed for the particular flight regime. Yet another option, called Speed Intervene, allows the pilot to override the Flight Management Computer input and *manually* enter a different speed by pressing the speed knob and then dialing in the desired speed to the Mode Control Panel.

Figure 1 is a modeling structure of an interface mode. It has three concurrently active processes (separated by a broken line): speed knob behavior, speed knob indicator, and speed window display. The behavior of the speed knob (middle process) is either "Normal" or "Pushed-in." (These two states are depicted, in the Statecharts language, by two rounded rectangles). The initial state of the speed knob is Normal (indicated by the small arrow above the state), but when momentarily pushed, the speed knob engages or disengages the Speed Intervene sub-mode of the Vertical Navigation (VNAV) mode. The transition between Normal and Pushed-in is depicted by a solid arrow and the label "Push" describes the triggering event (we shall describe the */b* later). The transition back to Normal occurs immediately as the pilot lifts his or her finger (the knob is spring loaded).

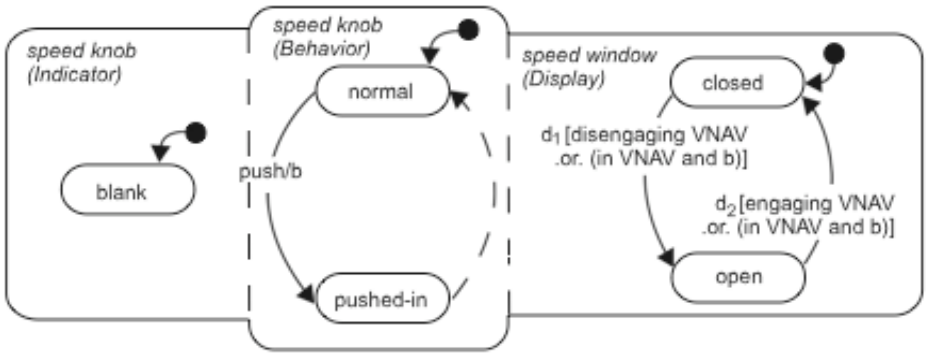

Figure 1. Interface Mode

The left-most process shown in Figure 1 is the speed knob indicator. In contrast to many such knobs that have indicators, the B-757 speed knob itself has no indicator and therefore is depicted as a single (blank) state. The right-most process is the speed window display that can be either closed or open. After VNAV is engaged, the speed window display is closed (implying that the source of the speed is from another component—the Flight Management Computer). After VNAV is disengaged, and a semi-automatic mode such as Vertical Speed is active, the speed window display is open and the pilot can observe the current speed value and enter a new speed value. This logic is depicted in the speed knob indicator process in Figure 1: transition d1 from Closed to Open is conditioned by the event "disengaging VNAV," and d2 will take place when the pilot is "engaging VNAV."

When in Vertical Navigation mode, the pilot can engage the Speed Intervene sub-mode by pushing the speed knob. This event, "push," (which can be seen in speed knob behavior process) triggers event b, which is then broadcast to other processes. Being in VNAV and sensing event b ("in VNAV and b") is another (.OR.) condition on transition d1 from Closed to Open. Likewise, it is also the condition on transition d2 which takes us back to Closed. To this end, the behavior of the speed knob is circular; the pilot can push the knob to close and push it again to open, *ad infinitum*.

As explained above and seen in Figure 1, there are two sets of conditions on the transitions between Close and Open. Of all these conditions, one—namely "disengaging VNAV"—is not always directly within the pilot's control; it sometimes takes place automatically (e.g., during a transition from VNAV to the Altitude Hold mode). Manual reengagement of VNAV will cause the speed parameter in the speed window to be replaced by the economy speed computed by the Flight Management Computer. If the speed value in the speed window was a restriction required by ATC, the aircraft will now accelerate/decelerate to the computed speed and the ATC speed restriction will be ignored! (See Degani & Kirlik, 1995, for an analysis of such documented incidents).

Functional Modes

When we survey the use of modes in devices, an additional type of mode emerges–the *functional mode*–which refers to the active function of the machine that produces a distinct behavior (Johnson, 1990; Cook, Potter, Woods, and McDonald 1991). An automatic gearshift mechanism of a car is one example of a machine with different modes

(e.g., Neutral, Reverse, Drive), each one defining different behaviors (idle, backward, forward).

As we move into discussion of functional modes and their use in machines that control a timed process, we encounter the concept of *dynamics*. In dynamic control systems, the configuration and resulting behavior of the machine are a combination of a mode and its associated parameter (e.g., speed, time) (Lambregts, 1983). Referring back to our car example, the active mode is the engaged gear (e.g., Drive) and the associated parameter is the speed that corresponds to the angle of the accelerator pedal (e.g., 65 miles per hour). Both mode (Drive) and parameter (65 miles per hour) define the configuration of the mechanism.

Figure 2 depicts the structure of a functional mode in the dynamic automated control system of a modern airliner. Two concurrent processes are depicted in this modeling structure: (1) modes, and (2) parameter sources.

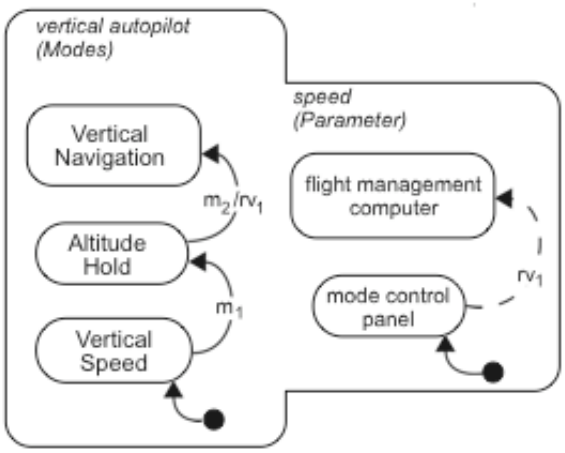

Figure 2. Functional Mode

Three modes are depicted in the vertical modes super-state in Figure 2: Vertical Navigation (VNAV), Altitude Hold, and Vertical speed (the default mode). All are functional modes related to the vertical aspect of flight. The speed parameter can be obtained from two different sources: the Flight Management Computer or the Mode Control Panel. The default source of the speed parameter (indicated by the small arrow) is the Mode Control Panel. As mentioned in the discussion on interface modes, engagement of Vertical Navigation via the Mode Control Panel will cause a transition to the Flight Management Computer as the source of speed. This can be seen in Figure 2 where transition m2 will trigger event rv1, that, in turn, triggers an automatic transition (depicted as a broken line) from “mode control panel” to “flight management computer.”

In many dynamic control mechanisms, some mode transitions trigger a parameter source change while others don’t. Such interdependency appears to be a consistent source of confusion to operators (Degani, 1996, Chap. 5).

Supervisory Modes

The third type of mode we discuss here is *supervisory modes* (sometimes also referred to as "participatory" or "control" modes (Wickens & Kessel, 1979; Kirlik, 1993). Modern automated control mechanisms usually allow the user flexibility in specifying the level of human and machine involvement in controlling the process (Sheridan, 1992). That is, the operator may decide to engage a manual mode in which he or she is controlling the process; a semi-automatic mode in which the operator specifies target values, in real time, and the machine attempts to maintain them; or fully automatic modes in which the operator specifies in advance a sequence of target values (i.e., parameters) and the machine executes these automatically, one after the other.

Figure 3 is an example of a supervisory mode structure that can be found in many control mechanisms, such as the automated flight control system, cruise control of a car, and robots on assembly lines. The modeling structure consists of hierarchical layers of super-states, each with its own set of modes.

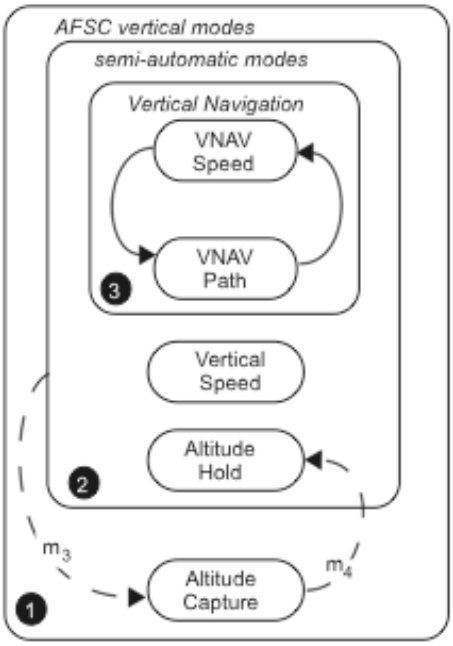

Figure 3. Supervisory Mode

The supervisory modes in the Automated Flight Control System (AFCS) are organized hierarchically. Three main levels are described in Figure 3. The highest level of automation is the Vertical Navigation mode (level 3), depicted as a super-state at the top of the mode pyramid. Two sub-modes are encapsulated in the Vertical Navigation mode—VNAV Speed and VNAV Path—each one exhibiting a somewhat different control behavior. One level below (level 2), are two semi-automatic modes: Vertical Speed and Altitude Hold.

One mode in the AFCS, Altitude Capture, can *only* be engaged automatically—no direct manual engagement is possible. This mode engages automatically when the aircraft is beginning the level-off maneuver to capture the selected altitude. When the aircraft is several hundred feet from the selected altitude, an automatic transition from any climb mode (e.g., Vertical Navigation or Vertical Speed) to Altitude Capture takes place (m3). Finally, when the aircraft reaches the selected altitude, a transition back from Altitude Capture to Altitude Hold mode also takes place automatically (m4)

In summary, we have illustrated a modeling language, Statecharts, for representing human interaction with control systems, and proposed a classification of three different types of modes that are employed in computers, devices, and supervisory control systems: *Interface* modes that change display format (e.g., Text and Command modes in a word processor). *Functional* modes allow for different functions (e.g., Reverse, Neutral, and Drive in a gear shift mechanism of a car) and their associated parameters (e.g., speed). Last are supervisory modes that specify the level of supervision (manual, semi-automatic, and fully automatic) in the human-machine system (but see Leveson, Pinnel, Sandys, Koga, & Reese, 1997, for another classification scheme). The three types of modes described here are essentially similar, in that they all define the manner in which a certain component of the machine behaves. The component may be the interface only, a function of the machine, or the level of supervision. This commonality brings us back to our general working definition of the term "mode"—a machine configuration that corresponds to a unique behavior.

SUMMARY AND CONCLUSIONS

In this paper our focus has been on the features of the machine that may lead to mode confusion and error in user-machine interactions. On the machine side we have discussed the constructs of mode behavior and mode ambiguity. On the user side we have discussed user expectations, which are influenced by three factors: the user's task, the user's knowledge of the machine's behavior, and the user's ability to sense triggering events.

But no systematic evaluation of human interaction with automated control systems can take place unless some representation is available. The building blocks of the representation suggested in this paper are the Statecharts language and the three modeling structures (interface, functional, and supervisory). The representation is oriented toward the user's perspective (but it can be refined to include more detail).

Now we can combine our understanding of the constructs that lead to mode error with a formal description of control systems. The first question that can be asked is whether the display allows the user to discern between machine configurations that are part of the user's task. Second, one can compare the user's model of the machine's behavior with these configurations, and ask whether the user's model makes it possible to predict the next configuration of the machine. Last, one can ask whether the display provides the user with all the information necessary to reliably predict when a transition will take place. Such analysis can highlight mismatches between the behavior of the machine and the corresponding information provided to the user via the interface (e.g., display) and training materials (e.g., manuals).

We believe that the mode constructs described in this paper and a State Machine modeling language such as Statecharts, provide the foundation and basic tools for systematic and formal evaluations of human interaction with control systems. The continuing search, we argue, must be for a systematic and formal analysis that will allow designers to identify the potential for mode error early in the design phase.

In conclusion, most complex control systems include modes, and their use will only increase in the foreseeable future. Mode confusion and error contribute to mishaps and fatalities in the operation of these systems. Only a combination of methodologies, some from very different disciplines, can provide us with effective resources to address this problem. This is our challenge.

ACKNOWLEDGMENTS

This work was conducted as part of NASA research on human interaction with automation. The first author was supported by Grant NCC-2-798 between San Jose State University and NASA Ames Research Center. An earlier version of this paper, titled "On the types of modes in human-machine interactions," was presented at the Ninth International Symposium on Aviation Psychology, Columbus, Ohio (April 28- May 1, 1997). The authors benefited from many discussions with Todd Callantine and Michael Heymann. Their contribution to the discussion of mode constructs described in the first part of this paper was invaluable. We thank Kenneth Funk, Rowena Morrison, and three anonymous reviewers for their constructive comments on an early manuscript.

REFERENCES

Abbott, K., Slotte, S. M., & Stimson, D. K. (1996). *The interface between flightcrews and modern flight deck systems*. Washington, DC: Federal Aviation Administration.

Ashby, R. W. (1956/1964). *An introduction to cybernetics*. New York: Methuen.

Billings, C. E. (1996). *Human-centered aviation automation: Principles and guidelines* (NASA Technical Memorandum 110381). Moffett Field, CA: NASA Ames Research Center.

Cook, R. I., Potter, S. S., Woods, D. D., & McDonald, J. S. (1991). Evaluating the human engineering of microprocessor-controlled operating room devices. *Clinical Monitoring, 7*, 217-226.

Degani, A. (1996). *Modeling human-machine systems: On modes, error, and patterns of interaction*. Unpublished doctoral dissertation, Georgia Institute of Technology, Atlanta.

Degani, A., & Heymann, M., (2000). *Formal aspects of human-automation interaction* (NASA Technical Memorandum 209600). Moffett Field, CA: NASA Ames Research Center.

Degani, A., & Kirlik, A. (1995). Modes in human-automation interaction: Initial observations about a modeling approach. *Proceedings of the IEEE International Conference on Systems, Man, and Cybernetics* (pp. 3443-3450). Vancouver, Canada.

Gentner, D., & Stevens, A. L. (1983). *Mental models*. Hillsdale, NJ: Lawrence Erlbaum.

Harel, D. (1988). On visual formalisms. *Communications of the ACM, 31(5)*, 514-530.

Jacob, R. J. K. (1986). A specification language for direct-manipulation user interface. *ACM Transactions on Graphics, 5(4)*, 283-317.

Kirlik, A. (1993). Modeling strategic behavior in human-automation interaction: Why an "aid" can (and should) go unused. *Human Factors, 35(2)*, 221-242.

Johnson, J. (1990). Modes in non-computer devices. *International Journal of Man-Machine Studies, 32*, 423-438.

Johnson, J., & Engelbeck, G. (1989). Modes survey results. *SIGCHI Bulletin, 20(4)*, 38-50.

Lambregts, A. A. (1983). *Operational aspects of integrated vertical flight path and speed control system* (SAE technical paper 831420). Warrendale, PA: Society of Automotive Engineers.

Leveson, N. G., Pinnel, L. D., Sandys, S. D., Koga, S., and Reese, J. D. (1997). Analyzing software specifications for mode confusion potential. *Proceedings of the Workshop on Human Error and System Development Conference*. Glasgow, Scotland.

Mett, P., Crowe, D., & Strain-Clark, P. (1994). *Specification and design of concurrent systems*. London: McGraw-Hill.

Monk, A. (1986). Mode errors: A user-centered analysis and some preventative measures using keying-contingent sound. *International Journal of Man-Machine Studies, 24*, 313-327.

Norman, D. A. (1981). Categorization of action slips. *Psychological Review, 1(88)*, 1-15.

Poller, M. F., & Garter, S. K. (1984). The effects of modes on text editing by experienced editor users. *Human Factors, 26(4)*, 449-462.

Reason, J., & Mycielska, K. (1982). *Absent minded? The psychology of mental lapses and everyday error*. Englewood Cliffs, NJ: Prentice-Hall.

Sheridan, T. B. (1992). *Telerobotics, automation, and human supervisory control.* Cambridge, MA: MIT Press.

Sherry, L., and Ward, J. (1995). A formalism for the specification of operationally embedded reactive systems. *Proceedings of the 14th AIAA/IEEE Digital Avionics Systems Conference*. Cambridge, MA.

Stenning, K., & Oberlander, K. (1995). A cognitive theory of graphical and linguistic reasoning: Logic and implementation. *Cognitive Science, 19(1)*, 97-140.

Tesler, L. (1981). The SmallTalk environment. *Byte, 6(8)*, 90-147.

Thimbleby, H. (1982). Character level ambiguity: consequences for user interface design. *International Journal of Man-Machine Studies, 16*, 211-225.

Thimbleby, H. (1990). *User interface design.* Wokingham, England: Addison-Wesley.

Wickens, C. D., & Kessel, C. (1979). The effect of participatory mode and task workload on the detection of dynamic system failures. *IEEE Transactions on Systems, Man, and Cybernetics, 9(1)*, 24-34.

Woods, D., Sarter, N., & Billings, C. (1997). Automation surprises. In G. Salvendy (Ed.), *Handbook of human factors and Ergonomics* (pp. 1926-1943). New York: John Wiley.